Spirit of Berlin: An Autonomous Car for the DARPA Urban Challenge Hardware and Software Architecture

Team Berlin
June 1st, 2007

Dr. Javier Rojo
Rice University, 6100 Main, Houston, Texas
jrojo@rice.edu

Dr. Raúl Rojas
Freie Universität Berlin, Germany, and Rice University, Houston, TX
rojas@inf.fu-berlin.de

Ketill Gunnarsson, Mark Simon, Fabian Wiesel, Fabian Ruff, Lars Wolter, Frederik Zilly, Neven Santrač, Tinosch Ganjineh, Arash Sarkohi, Fritz Ulbrich, David Latotzky, Benjamin Jankovic, Gretta Hohl
Freie Universität Berlin, Germany

Thomas Wisspeintner, Stefan May, Kai Pervölz, Walter Nowak, Francesco Maurelli, David Dröschel
Fraunhofer Gesellschaft (IAIS), Sankt Augustin, Germany

www.spiritofberlin.eu

Executive Summary

Spirit of Berlin is a conventional vehicle modified for drive-by-wire using commercial technology available for handicapped drivers. For global wide-area navigation, the car relies on a GPS/IMU system that provides 1 meter, or better, of positioning accuracy. The car adjusts its driving behaviour using local information: lane markings are detected using video cameras; obstacles are sensed using a frontal long-range (150 m) laser scanner, and a secondary scanner mounted in the back of the car. Information from the laser sensors is used to provide a correction for GPS positioning using particle filter scan-matching, a discrete form of the Kalman filter. We have also developed a novel low-cost 3D laser scanner that uses two SICK single-beam scanners rotating around a vertical axis.

Our system is *lean* – it integrates industrial quality moderate-cost sensors and actors. The software is *robust* because our robotics framework is checking processes continually. The control hardware has *built-in redundancy*. Sensor fusion is performed by the software. High-level behaviour control is realized by a planner which computes optimal paths, specially around obstacles. Reactive control components guarantee that the car can react safely to unexpected situations. A simulator allows testing and development of adaptive controllers using machine learning methods.

I Introduction and Overview

1 Team Berlin

Team Berlin is a joint team of researchers and students from Freie Universität Berlin, Rice University, and the Fraunhofer Society (Institute for Intelligent Analysis and Information Systems). The team leader, Prof. Javier Rojo, is a faculty member at Rice. Our second team leader, Prof. Raúl Rojas, is a faculty member at Freie Universität Berlin and visiting professor at Rice University. The rest of the team is composed of graduate and undergraduate students from FU Berlin, Rice University, as well as researchers from the Fraunhofer Society in Sankt Augustin, Germany. Team Berlin started in 2006 following a three month visit of some of our team members to Stanford University. We worked in the group of Prof. Sebastian Thrun, winner of the Grand Challenge 2005 [Thrun 05], where we had the privilege of witnessing the birth of Stanford's entry to the Urban Grand Challenge. The members of the Stanford team were very helpful and gracious in providing information and technical papers that provided the motivation for our own team.

At Freie Universität Berlin we have been building autonomous robots since 1998 [Behnke 00]. Over the years, we have built several generations of mobile machines which take part in the annual RoboCup (robotic soccer) competitions. Our team won the World Championship in the small-size league twice, and won second place in the middle-size league once. We dominated the European competitions over several years, winning the European championship five times.

In 2006, encouraged by our success with fast autonomous robots equipped with computer vision [Hundelshausen 01], we decided to start building larger robots. Our decision to enter DARPA's Urban Challenge was based on a consideration of the areas in which autonomous robots could play a role in the not so distant future. By coincidence, the Berlin police department approached us, at about the same time, asking us to develop a security robot for large warehouses and enclosed areas (such as airports). We proposed to develop and autonomous car and seed funding from FU Berlin was granted for the project. The system had to be lean and the total cost

had to be kept under a restrictive ceiling, so that the autonomous car could be deployed in the near future. Sensors inaccuracies would have to be counterbalanced using sensor fusion and statistical methods.

In October of 2006, a Dodge Caravan was purchased. The car had been modified so that a handicapped person could drive using a linear lever for brake and gas (the lever controls all intermediate steps between full braking and full acceleration), and a small wheel for steering the front wheels. Other car's components are controlled through a small touch-sensitive panel. Electronic Mobility Controls (EMC) in Louisiana installed additional electronics so that the car can now be controlled using a computer connected to special converters. EMC also installed the interface for the E-Stop-Signal. The car was shipped from Houston to Berlin where the bulk of the hardware and software development has been done. The car was shipped back to Houston on May 18, where development at Rice University continues.

Per DARPA specifications the Urban Challenge consists in driving a car, autonomously, in a city environment whose street network has been captured in a sparse graph of GPS points. Four-way crossings, static, and dynamic obstacles are present. Stop signs have to be respected, as well as precedence traffic rules. The car should be able to take an alternative route when a street is closed, doing a U-turn if necessary, and should be able to navigate and park in unstructured parking lots. It is not necessary to recognize traffic lights or traffic signs, nor pedestrians. These specifications simplify the driving challenge, which, nevertheless remains formidable since 60 miles have to be covered under stringent time limits.

2 The robotic platform

Choosing a robot

Selecting the robotic platform was the first design issue confronted by our team. Many new car models already allow controlling accelerating, braking, and steering via CAN bus commands. However, most of this information is proprietary and the companies we approached (for example Volkswagen) charge in excess of $150,000 for a drive-by-wire vehicle. To reduce the cost, taking advantage of available technology, we decided to look at vehicles for the handicapped. The technology and necessary actuators have been available for almost 35 years. It is a mature segment of the car industry offering vehicles which can drive for thousands of miles without hardware errors. We bought the car in October of 2006 for $25,000. It is a Dodge Grand Caravan modified for drive-by-wire by EMC. The company also provides an interface for computer control of the EMC electronics. We have not had a single problem with the car's actuators or computer commands. The robotic platform is stable and reliable.

Having a vehicle that obeys computer commands (accelerating, braking, steering, as well as controlling lights and other elements of the car) solves just one aspect of the hardware problem. The next major issue is providing enough energy for additional sensors and control computers. The sensors provide the perception of the environment, which is then used by the computers to produce the necessary control commands.

2.1 Vehicle control interface and power system

Electronic control

Fig. 1 shows the vehicle's control and power system. The drive-by-wire actuators are operated by an embedded computer with three kinds of interface channels: a) an analog input connection is used for steering and a separate one for accelerating or braking; b) a serial input connector accepts commands for changing gears, controlling the car's lights, and other secondary functions;

c) a CAN-bus interface is used to retrieve real-time information about the car's speed, the current position of the steering wheel, and other data. All three interfaces are connected to modules that provide an Ethernet interface to the control-computers (e.g. Ethernet-to-Serial, Ethernet-to-CAN and Ethernet-to-Analog). The car can be driven autonomously sending commands from the control computers to the embedded computer, or it can be driven by a person using manual controls (joystick and wheel). For security reasons, we test our vehicle with an operator in the driver's seat. The operator can disengage autonomous driving by pressing a button and can take full control of the vehicle using the manual drive controls.

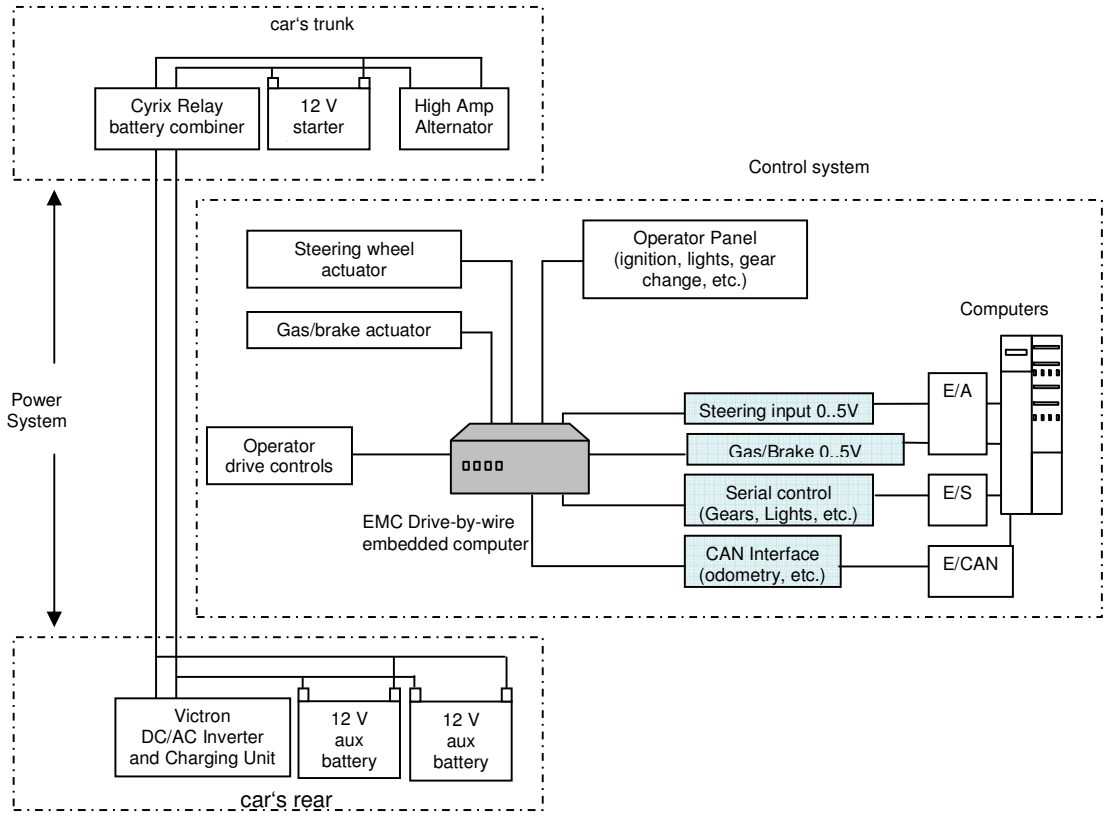

Figure 1: Diagram of the control and power system. An alternator delivers energy to the car battery, and to two additional buffer batteries for the sensor and control electronics. An embedded computer operates the car's actuators and responds to the commands sent by the mission control computers.

Figure 2: Two of the EMC drive-by-wire components. The manual console for the operator (left image). View of the motor for braking or accelerating (right image). Accelerating and braking are mutually exclusive operations.

Providing power

Providing the necessary energy for the computers and sensors is a major challenge in a conventional car. In order to have enough power, a new alternator with a maximum output of 200-250 Amperes, and 140 Amperes, when the motor is idling was installed. We also mounted new power distribution cables in the car. After the drive-by-wire system and other OEM-equipment have had their share of power, we still have 1000 watts to accommodate the power needs of computers and sensors. The two additional batteries mounted in the back of the vehicle can supply the power demands of computers and sensors for a few hours. Such power buffering allows the seamless operation of the control system regardless of motor speed.

2.2 Safety

E-Stop

As mandated by DARPA for the Urban Challenge, our car can be activated and stopped using a remote control. It can also be stopped by pressing any of the stop buttons located on each side of the vehicle. The E-Stop signal is a single bit which is sent to a receiver connected to the EMC embedded computer. On reception of the signal, the embedded computer stops the car safely and quickly. The stop signal can be disabled pressing another button in the remote control, and the car resumes its autonomous drive.

Figure 3: Safety: The E-Stop remote control unit can stop and restart the car remotely. A safety driver in the cockpit can regain control by pushing a button. The brakes can always be used.

2.3 Computer architecture and sensor connections

Control hardware architecture

After settling on a robotic platform, the next design issue is the hardware architecture of the control and sensor systems. Although in the past we have always used a single computer in our mobile robots, for this project we decided to adopt a symmetrical computing approach. Each computer (two at the moment, but the number is variable) is connected to the sensors by either an Ethernet or an IEEE-1394 bus. The laser scanners, the embedded control computer, and the navigation system are connected to the Ethernet bus. Each computer is equipped with two Gigabit-Ethernet ports. This allows a redundant setup of the network through two independent switches, that is, we have two independent Ethernet networks. This architecture provides reliability and scalability, since any computer is capable of taking over the workload of another one. The architecture relies on a decentralized workload manager running in parallel in each computer which distributes the tasks depending on the capabilities or possible faults of each unit. Should the computing power be insufficient at any moment, the workload manager can provide graceful

degradation of the complete system, substituting sensors or even stopping the car whenever an unsafe condition is detected.

The video cameras are connected to the Firewire connections. Due to bandwidth requirements each camera needs its own separate IEEE-1394 bus connection. Every computer is equipped with a OHCI-Firewire-controller daisy-chained to all cameras. This allows any computer to access all cameras, but only one at a time. Access to a specific camera will not be impaired by the loss of either a computer or an OHCI-controller.

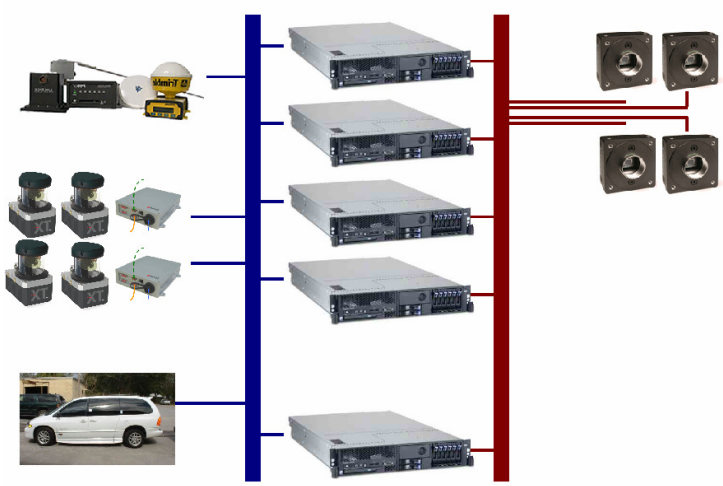

Figure 4: The sensor and control connections. Navigation system, laser scanners, and the embedded control computer are connected to the Ethernet bus. Video cameras are connected to a Firewire bus.

2.4 Operating system and software framework

Linux

A major design issue, from the beginning, was the choice of operating system and its associated real-time capabilities. We chose Linux because it provides the necessary flexibility for the project. It is easy to install across different hardware platforms and has the necessary responsiveness for our soft real-time needs. It can be tailored to our requirements, stripping it down to the bare minimum.

Our software framework is based on the *Orocos Realtime Toolkit*. Orocos stands for *Open Robot Control Software* – it is open source software for robot and machine control based on C++ libraries [Bruyninckx 07]. In conjunction with RTAI (Real Time Application Interface) or Xenomai, it can be used to provide strict real-time services for Linux. Orocos depends on CORBA for distributed computing purposes.

In Orocos, the software is divided into modules, called TaskContexts, that communicate over software ports, thus providing lock-free and thread-safe data exchange. A module can be either executed periodically or can be triggered by external events, such as new GPS data or a new video frame. It can be configured through “properties” and commands. All this functionality is made available transparently over the network via CORBA proxy objects.

We decided to model sensors in our framework as abstract modules which react to external events. A sensor module time-stamps the data accessing a globally synchronised clock, and forwards the raw data to a port for further processing by a module.

The perception software has been implemented as periodic access modules which poll the provided data, process it, and update their corresponding models. For example, a lane detection

module can poll a camera-module for new images, and then process the image to find lane markers. The detected lines are then used to update a particle filter, which is a model used by the navigation system. By interfacing modules through ports, we can easily distribute the tasks across several computers since each port can transparently serialize the data via CORBA's IIOP (Internet Inter-Orb Protocol). Using this abstraction and providing all computers with the same set of modules, a workload manager (TaskManager) is free to choose which tasks to run.

Health monitor

In a complex system, such as an autonomous car, many software modules and hardware sensors are in continuous interaction. Failures can be catastrophic. To insulate the system from partial failures, and to make it more resilient, we have designed a supervising piece of software which we call the "health monitor". Activities and connections between devices and other modules are checked continuously. We check the input connections from laser-scanners, cameras, and the navigation system, and the output connections to the brake and gas pedals, as well as to the steering wheel. Activities and connections are considered healthy if they pass a set of tests. Each socket, for example, is tested periodically by requesting an echo. Orocos data ports and software modules are checked using built-in methods provided by the Orocos framework

Data exchanged between modules is always examined by checking the age of time-stamped data to detect inactive devices or modules. In addition, data-specific values are tested against reference values to identify abnormal behaviour. Other checks include the comparison of data obtained from different modules. We can check the relationship between the acceleration command, the expected speed, and the actual velocity of the car (factoring out the delay). When unhealthy activities or connections are detected (those which do not pass the tests), the corresponding modules or devices are reset or restarted via specific Orocos methods, Ethernet-commands or, in extreme cases, by switching off and on the offending component.

2.5 Sensor Architecture

Once the car's control electronics is defined and the computing power demands are satisfied, the third element necessary for autonomous driving is a set of sensors. While humans can drive guided almost exclusively by their sense of vision and a coarse mental map of a city, autonomous cars can access much more precise information about the environment but have reduced computer vision capabilities. A long term goal of AI research is to be able to emulate the object recognition ability of the human brain but we are far from reaching that ultimate goal. Therefore, the shortcut adopted by almost all teams developing autonomous cars has been to provide the vehicle with the following sensors:

- an accurate GPS navigation system
- powerful short and long range laser scanners, both flat and 3D
- radar/sonar
- lasers for reflectivity measurements of the ground
- special purpose computer vision

Sensors

Spirit of Berlin is no exception. One immediate design decision, after having a drive-by-wire vehicle, was to acquire a GPS navigation system with an inertial unit capable of providing better than 1 meter accuracy. The next decision was to acquire an Ibeo long range laser scanner for detecting obstacles at up to 150 meters in front of the vehicle. Subsequent decisions were, by comparison, easy and straightforward: we bought an inexpensive laser scanner for the rear of the vehicle and mounted several combinations of video cameras on top of the car (one omnidirectional camera and two cameras for stereo vision). It was also decided to develop a 3D scanner

composed of two rotating SICK LIDARs. Our sensor architecture depends strongly on the accuracy of the navigation system and the obstacle detection capabilities of the laser scanners. We decided to use the computer vision system mainly for segmenting the street and for finding curb and lane delimiters. We are currently working on the detection of vehicles using stereo vision, this feature will be available until after the site visit. Most Grand Challenge teams participating in 2005 had limited computer vision and were heavily dependent on GPS navigation and LIDARs [Dahlkamp 06]. In the coming years we will focus our efforts on making computer vision more dependable, because such an advance in the pattern recognition field would make autonomous cars affordable.

2.6 Global Navigation

Global navigation is done in our system using an Applanix LV220 GPS navigation system. The system provides a positioning accuracy of 1 meter using differential GPS, and better than 10 cm using inertially aided RTK. During a 2 Minute GPS outage the accuracy of the system, which includes an inertial unit, can degrade up to 1 meter accuracy. The navigation system communicates with the control computer using an Ethernet connection. The LV220 system works with two antennas, providing the user not only the three coordinate position of the car but also the heading. The pitch, yaw, and roll of the car are provided by the Inertial Management Unit. These measurements are important for matching laser range scans and building a world model.

Commercial navigation system

Choosing the appropriate navigation system was an important design decision. Here, we decided to follow the advice of the Stanford team, who in 2005 wrote their own Kalman filter and built an integrated GPS/IMU navigation system. Their measurements and experience showed that a commercial system such as the Applanix offers better performance. Therefore, we decided to spend development time in other areas and work with a commercially available high-grade navigation system.

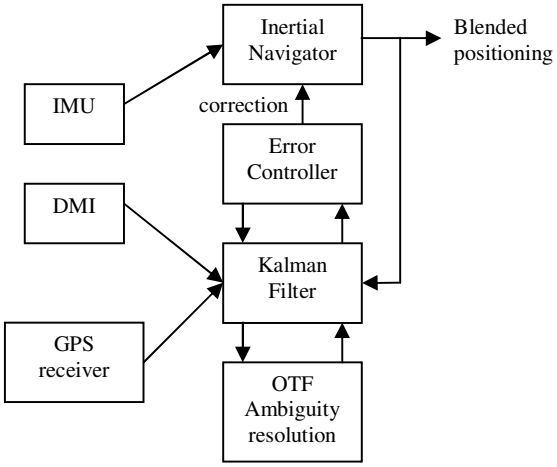

Figure 5: The Applanix LV220 correction loop. A Kalman filter integrates odometry and GPS producing a correction signal for the inertial unit. Ambiguities are handled on the fly [Scherzinger].

The Applanix LV220 system consists of a processing computer, a GPS receiver, an IMU based on ring laser gyro technology, and a DMI (distance measurement indicator) which is an odometer attached to one of the rear wheels. The GPS receiver can use free or commercial differential correction services. Fig. 5 shows the dataflow in the complete system. The odometry and raw

GPS data are fusioned using a Kalman filter and a model of the car. This data is used to correct the information from the IMU, eliminating drift. The Kalman filter operates with the raw data of one or more satellites, providing positioning information even when the GPS signals are shaded.

2.7 The simulator

Virtual testbed

Testing a robot or an autonomous car is time consuming and expensive. Therefore, it was clear from the beginning that a simulator is needed as a virtual platform for testing and developing control algorithms. We simulate the vehicle dynamics using a coarse physical approximation of the car behaviour. For the dynamics of the vehicle we build upon E. Rossetter's PhD thesis [Rossetter 03]. Our measurements have shown that we do not need a full fledged simulation of tire behaviour and aerodynamics within the limited range of velocities we have been working with until now. It is far more important to handle the latencies in steering and acceleration, as well as the change in acceleration for the same pedal position when different gears are engaged. The basis for our estimations are a number of tests with the real car which give us a characterization of the system so that the simulator behaves as closely as possible to the real vehicle.

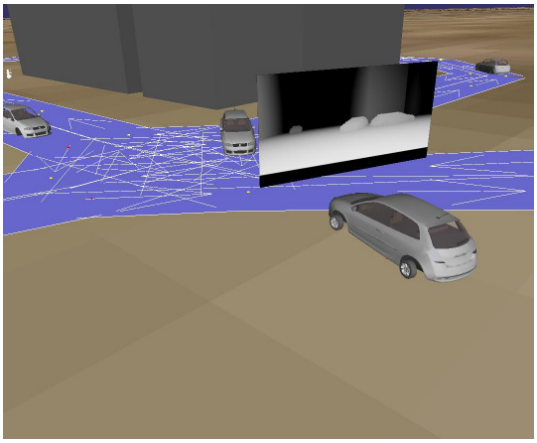

Figure 6: A view of the 3D control simulator.

To test the high-level behaviour of the vehicle, we recreate the 3D surroundings of the car. The simulator can handle static as well as dynamic obstacles. A simulation file, together with an RNDF of the area being considered, serve to create a living world where other cars drive and humans walk the sidewalks or cross the streets. For each dynamic object in the world we generate a simulated agent that in case of a car, for example, just follows the roads in the streets graph.

The simulator works with a graph generated from the RNDF which allows us to easily perform shortest path searches and to handle neighbourship relationships between roads, crossings, etc. This graph is also used by high-level behaviour control for planning operations, and is extended and corrected (in case of a closed road, for example) by sensor modules.

Simulating sensors

The simulator provides graphical output. This output is used to observe the results of new algorithms. It can render data generated by all the different modules together with textual information attached to objects or locations. Objects can be selected and changed allowing, for example, graphical real time editing of the RNDF graph.

The simulator can also generate the data which would be obtained from cameras and laser scanners mounted on a vehicle. In Fig. 6 the grey level screen in front of the vehicle represents

the distance of the car to all visible objects. Bright pixels are close; the darker they get the farther away the corresponding object is. We can also add some noise to the generated data in order to test the laser scanning algorithms under unfavourable conditions.

II Analysis and Design

3 Distance Sensors

Sensors are essential for detecting obstacles and to measure the smoothness of the road, for the identification of road lanes, and even for navigating in the absence of a GPS signal (for example in tunnels). We have installed two main types of sensors in our car: laser scanners based on time-of-flight measurements, and video cameras, omnidirectional as well as an stereo pair. We are also developing a 3D-laser scanner for the detection of obstacles mainly at streets crossings. In the following sections we review the various types of sensors and the perception algorithms.

3.1 Frontal four-beam laser scanner

LIDARs are popular sensors in the field of autonomous robots. The principle is well-known: a laser diode provides a beam of light which is deflected by a rotating mirror. The deflected beam scans an area in front of the device (for example, horizontally from left to right) and the time of return of the deflected beam is measured. The time of return is proportional to the distance of the reflecting object. In our car we have installed SICK LMS 291-S05 laser scanners, and one Ibeo Alasca XT unit with four laser beams.

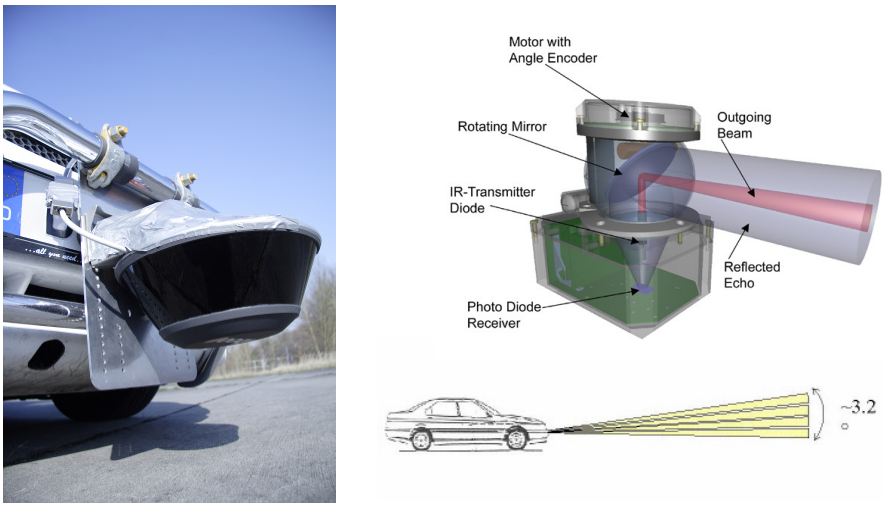

Figure 7: The Ibeo laser scanner mounted in front of our vehicle (left image). On the right we see a diagram of the unit and the angles covered by the four laser beams (courtesy of Ibeo).

Fig. 7 shows the Ibeo laser scanner mounted in the front of our car, as well as a diagram of the unit. The Ibeo electronic control unit (ECU) inside the car receives the signals from the four

beam detectors. The ECU is an embedded computer running Windows XP Embedded. The laser scanner provides four scan measurements at different vertical angles, with a total opening angle of 3.2 degrees. We operate the scanner at a frequency of 12.5 Hz resulting in an angular resolution between scan points in the same plane of about 0.5 degrees. While the scanner supports a field of view of up to 270 degrees, our current mounting setup yields a 220 degree field of view in front of the car. The Ibeo specifications list a range of sight of the laser scanner of up to 200 meters. We have obtained good results at up to 150 meters.

150 m range

The ECU connected to the laser scanner provides complete detection and tracking of objects, as well as the raw scan data. The object data structure includes a contour of the object, a classification (pedestrian, bike, car, truck), and speed and direction in the case of moving obstacles. For correct estimation, the ego-motion (velocity and steering angle) of our vehicle is fed to the ECU through a CAN bus.

Apart from using the built-in object detection of the Alasca system, we have developed an independent object detection solution based on the raw scan data provided by the ECU to our control computer. Our approach is to filter reflections from the ground due to rolling and pitching of the car when it is accelerating or braking, using information from the IMU. Thus, our obstacle detection software filters reflections from the ground.

The first step for obstacle detection is to cluster scan points. The scan points are examined sequentially from left to right and the Euclidian distance between two consecutive points is thresholded. Remember that we have four scan planes, so several points are direct neighbours. Points close to each other are assumed to belong to the same object. In the second step, the clusters which have been detected are matched to previous scans in order to track objects through time. Successful matches provide an estimation of the speed and direction of moving obstacles.

3.2 Rear LIDAR

In order to cover the back of the vehicle, we mounted a single beam IBEO-LD LIDAR (the predecessor of the Alasca) in the back of the vehicle. We have also used this LIDAR to scan the street in front of the vehicle to determine its flatness. The scanner has been setup to acquire a single plane with a 150 degree view at 0.25 degree resolution, at a scanning frequency of 10 Hz. We use all data within a 150 m range. Although the frequency is relatively low, it is enough to detect obstacles higher than the ground even if the car is moving at 30 mph, especially in drive-around-an-obstacle situations.

The mission control computers communicate with the scanner's embedded computer via an Ethernet link. The embedded computer, a Linux machine, communicates with the scanner through an ARCNet link. The sole purpose of the embedded computer is to filter the raw data.

3.3 The Fraunhofer 3D Scanner

The Fraunhofer Society (IAIS), collaborating with our team, has developed a new continuously rotating 3D laser scanner for our autonomous car. Two industrial laser range finders rotate around the vertical axis of the system, acquiring depth and intensity information across a 360° field of view. When mounted on top of the car, the system can be used to detect roads, streets crossings and obstacles. We make the detection process more reliable by fusing the depth and intensity information obtained by each LIDAR.

3D world view

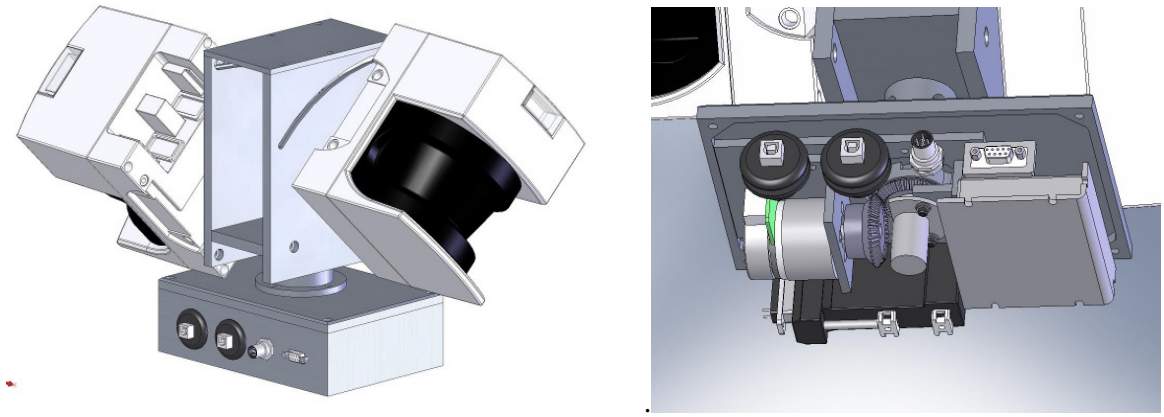

Fig. 8: CAD drawing of the 3D Scanner and of the motor/gear box.

The system consists of two SICK LMS 291-S05 laser range finders mounted on an adjustable plate (Fig. 8). The scanners have an apex angle of 180 degrees and a resolution 1 to 0.25 degrees. The response time to acquire one two-dimensional scan is from 13 to to 53 milliseconds, depending on the angular resolution. The maximum range of the scanners is 80 meters.

Adjusting the angle of the scan planes allows us to increase the scan resolution, or to increase the rotation speed while cropping the top and the lower part of the scanned sphere. The adjuster plate rotates continuously around the vertical axis. It is driven by a brushless Maxon flat EC 45 50W motor with an 89:1 gear. Three Hall sensors acquire position feedback from the motor. The Maxon EPOS P contains an integrated micro controller for speed control and positioning, operating at 33 MHz. The firmware of the motor controller starts the motor after power-on and sets a predefined velocity. It is possible to communicate with the firmware program over a RS232 interface to set the angular velocity and obtain positioning information.

In a rotating system, providing connections is a major issue. Contact rings are used to power the laser scanners with 24V DC, 0.9A, and to transmit the sensor data via RS485 from the scanners to the 2 USB plugs in the control box. For this, two RS485 to USB converters are applied. An RS232 interface allows us to set the rotation speed and to get feedback from an inductive proximity switch which is used for absolute positioning.

The entire system is IP65 water resistant, weights 13kg and measures 290×330×250mm. Power consumption is 2.2A at 24V DC. The scan resolution depends on the rotation speed and the angle of the scanners. At an angular adjustment of 60 degrees and 0.45Hz we obtain a vertical resolution of 0.5 degrees and a horizontal resolution of 1.7 degrees, with an update frequency of 0.9Hz (twice the rate of rotation).

A separate control PC running Linux is used for post processing of the laser scanner data. It uses the RS232 interface for communication with the motor controller and USB to acquire the data from the laser scanners. The Ethernet interface is used to communicate with the car's control system. The control PC constantly queries the 2D scanners for new scans and uses the position of the mounting plate shaft to allocate a two-dimensional scan slice to the global coordinate frame.

3.4 3D Reconstruction

We transform the scanned data into a 3D point cloud. The two rotating scanners acquire data arrays continuously and send them to the application software. The data from the scanner is represented in polar coordinates. The value in the data array represents distance and the index in the array represents the angle in degrees.

We first transform from polar to Cartesian coordinates, adding an offset defined by the position of the center of the entire system. We obtain 2D Cartesian coordinates for the scanning plane using the transformations $x = d \sin\alpha + 10$ and $y = d \cos\alpha$. Due to the angular tilt of the two scanners, we have to additionally transform the 2D points in the scan plane into 3D points in world coordinates. This is done in two steps: first rotating the points around the *x*-axis by an angle θ (in our case θ = 60°), then taking into account the rotation of the scanner from the zero position around the z-axis by an angle β. The two sets of transformations below produce the respective rotations:

$$x' = x \qquad x'' = x'\cos\beta - y'\sin\beta$$
$$y' = -y\sin\theta \qquad y'' = x'\sin\beta + y'\cos\beta$$
$$z' = y\cos\theta \qquad z'' = z'$$

The angle β changes during a single scan because the scanners are rotating. Synchronization for calculating the correct angle β is done by reading an inductive proximity switch and calculating the time needed for one complete turn to distribute the scans over time. A software thread accesses the switch. When a turn is completed, we reset a timer. The angle β is given by the current time divided by the time needed for a single turn, multiplied by 360 degrees.

Fig. 9 shows an example of a 3D point cloud in world coordinates with distances encoded by different colors. The shadow of the vehicle is visible as absence of scan points on the ground. Trees are visible above the car.

The above formulas can be applied directly if the system is not moving. If the system is moving we have to do additional trajectory corrections for mapping the point cloud to world coordinates.

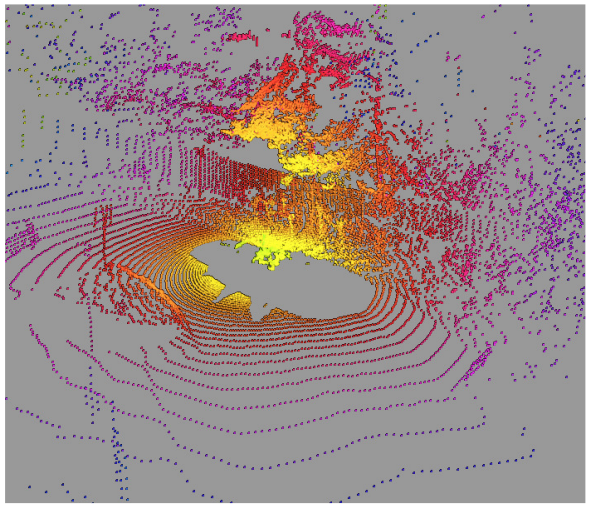

Figure 9: Reconstructed 3D cloud with distance values encoded by colors

3.5 The world model and GPS correction

One of the most interesting applications of the data provided by scanners, other than finding obstacles, is to produce correction values for the navigation system based on a world model. We accumulate the data of range sensing devices in a two-dimensional grid (Fig. 10). The cells side-length can be set to values between 20 and 30 cm. The grid data structure is optimized for fast access within the car's immediate surroundings. A 250 by 250 meter map requires about 64 MB of memory. A larger area can be covered with multiple smaller maps and using paging. Since the intended movement of the car along an RNDF is known in advance, paging the appropriate local map in advance can be done easily.

Various types of data are stored in each cell. The most important are the minimum and maximum height of detected obstacles, as well as the average and variance of the collected values. These numbers can be used for many purposes. For example, high variance of data in a cell may indicate the presence of a dynamic obstacle. Cells with static obstacles have more stable mean and variance. The most important application is providing navigation corrections.

The transformation of the data from relative coordinates to world coordinates is done using an affine transformation derived from the navigation data and the position and orientation of the lasers. We developed an automatic calibration technique which measures the relative roll/pitch and height of the laser relative to the ground. For this, we scan a cylinder at a given distance. Fitting ellipses to the data we can compute the orientation of the scan planes relative to the vertical and, from this, the pose of the laser scanner.

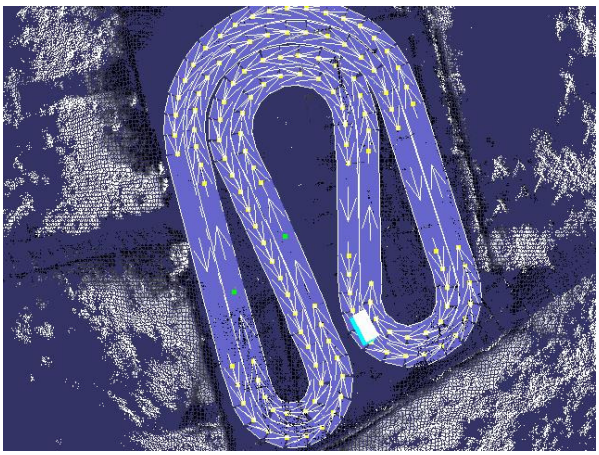

Figure 10: The accumulated world map after a test drive. White points represent obstacles, black points belong to the ground. The course track has been superimposed on the world map. Most of the obstacles are bushes and trees.

Particle filter for navigation corrections

The world model, built on the fly by the scanning software, can be used to provide navigation corrections in the presence of GPS outages or shading of the signal. We use a particle filter: for a given hypothesis of the car's position (a particle), we find an optimal movement of the data (and thus of the particle) in order to best match the current point-cloud of the laser scanners with the world map obtained from previous scans. When the car is moving at moderate velocities, the computed cloud of moving particles provides a good approximation to the probability distribution of the car's movement (Fig. 11).

The system is bootstrapped filling the two-dimensional grid as soon as scan data starts to arrive. The mean and variance of a world-matrix-cells become more accurate as more data is col-

lected. In the absence of moving obstacles, the computed variances and means are very precise. However, under normal conditions, we cannot assume that only static objects will be sensed, so the method has to be robust enough to cope with dynamic objects. Highly variable cells may indicate a dynamic object and have to be handled differently (in the matching/scoring function).

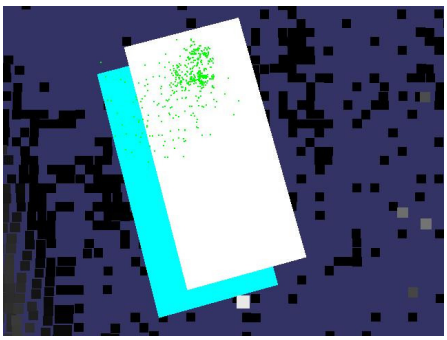

Figure 11: Visualization of the particle filter. The blue rectangle represents the current navigation data. The white rectangle represents the average of particle poses derived from scan matchings. The displacement between the two rectangles is the correction vector needed for the navigation system.

The matching-function is the mean of squared differences: for every particle pose, the current point cloud is translated by the particle's offset and the similarity with the world-map is determined using squared differences. Note that the current matching functions performs only a translation, which is enough to eliminate GPS drift. We are now extending the method to cover rotations which is important for handling GPS outage.

GPS outage

In the case of GPS outage, the GPS corrector can be used to localize the car in the world. After the GPS signal has been lost it takes some time for the car to globally localize itself again (especially if no valid GPS-position is present over a longer period of time). After the initial localization, poses may be computed in the vicinity, as in the GPS correction case, but considering also rotations of the vehicle's pose. This is SLAM in its beginning stage. The complexity of a matching step increases by a few dimensions (depending on whether we want to compute only the yaw of the car or also it's rotation and pitch). Each pose is thus extended from a two-dimensional vector (x,y) to a five-dimensional vector (x,y,yaw,pitch,roll). Once an initial translation and rotation of the car has been computed, the subsequent matchings become more precise since one can work within a smaller interval with a better resolution. The ensuing matching-steps are also faster as one does not have to scan all possible yaw/pitch/roll values but only those close to the current ones.

4 Computer vision

Distance sensors are very important for autonomous driving. However, humans derive all distance information from their sense of vision and their object recognition skills. This has proven to be very difficult to imitate using computers. Therefore, some shortcuts have been traditionally taken: the computer vision is usually limited to the recognition of colors, or to the recognition of very salient visual cues. We have been developing concurrently two computer vision systems for our autonomous car: one operates with a catadioptric omnidirectional system (a camera pointed towards a mirror), the other with cameras pointed towards the front of the vehicle. Here we review each one of the vision systems and their capabilities.

4.1 Omnidirectional vision

We have experimented with an omnidirectional camera consisting of a convex mirror which reflects the surroundings of the car when a camera is pointed vertically looking into the mirror upwards. We have successfully used such omnidirectional systems in the past [Hundelshausen 01]. Fig. 12 shows the omnidirectional view of the car's surroundings produced by the omnidirectional system. The white lines are visible, but detected white points have to be rectified as shown on the right of Fig. 12.

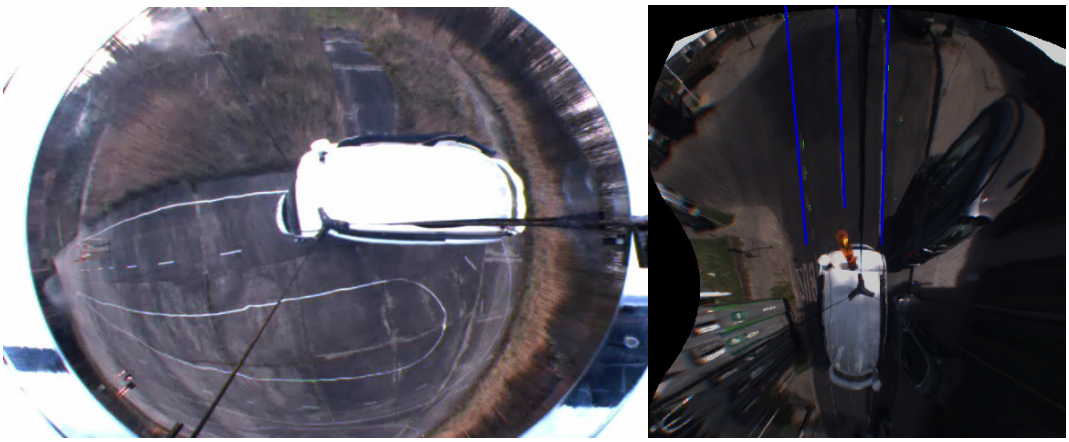

Figure 12: View of the car's surrounding using an omnidirectional camera. On the right, the white lane markers have been identified in a rectified image (the lane markers are highlighted in blue)

Our experience is that it is faster to process the information coming from conventional cameras, and that the resolution of the system is much better in that case. We started then using frontal cameras in our car, keeping the omnidirectional vision system as a backup.

4.2 Frontal camera

We use Firewire cameras on top of the vehicle, pointed towards the road ahead. We use them to identify white lane markers in order to center the vehicle in its lane while it is driving. RNDFs are sparse, and sometimes the real path between two way points has to be found using laser scanners or computer vision.

Finding white lane markers

Lane detection is done in three steps: a) first we find image pixels that might be part of a lane, b) using these points we formulate lane boundary hypotheses, c) we then transform from image to world coordinates. We review these steps one by one.

Lane markers candidate pixels are found by looking for intensity bumps of a certain width in every horizontal line. Since the lane marker width in a non-rectified image decreases with the distance from the camera due to perspective, we do not search bumps using a fixed pixel width. The width is instead a function of the depth at which we are searching lane markers. We assume that the road region in front of the car is flat. This algorithm is simple and very fast. Of course, false pixels can be recognized as lane markers and additional corrections are required in order to make lane detection robust.

We fit lines to the detected lane marking pixels using a Hough transformation with good results and in acceptable time. We achieve a frame rate of 19 to 20 fps on a 1,83 GHz Intel Core Duo machine with 1GB of RAM, close to the camera frame rate. The processor load including

the logger, the image conversion to RGB, lane detection, and the display of results is around 60% for both CPUs. We are now improving the efficiency and robustness of the algorithm by exploiting the temporal coherence between consecutive frames. Assuming that we have found the correct lane boundaries in the previous frame we can limit the search in the actual frame to a small region of interest, taking into account the motion of the car. We use a particle filter to minimize the probability of false lane detections. The Hough transformation will be replaced by a curve fitting algorithm based on RANSAC [Fischler 81] in order to better detect curved roads.

Once the lanes have been found in the image it is necessary to transform its image coordinates to world coordinates. For every relevant point (namely the start and end points of line segments), a projective transformation is computed. The system is calibrated in advance in order to find the correct coefficients of the transformation matrix, solving a system of linear equation mapping four or more image points to known points in the world. This is a conventional computation which we have implemented for our other mobile robots [Simon 04].

4.3 Stereo Vision

We have been experimenting with an stereo vision system which could be used to find white lane markers in streets and highways, but could also be used to determine the vehicle's pose independently from the IMU. We follow the techniques described by [Broggi 05].

The stereo vision system consists of two Firewire-cameras mounted on the top and front of the car. Both cameras share the same pitch and roll angle. The distance between the cameras and their yaw angle can be calibrated. The cameras have a dedicated trigger input port, so we can guarantee that the images of the left and right cameras get identical time-stamps.

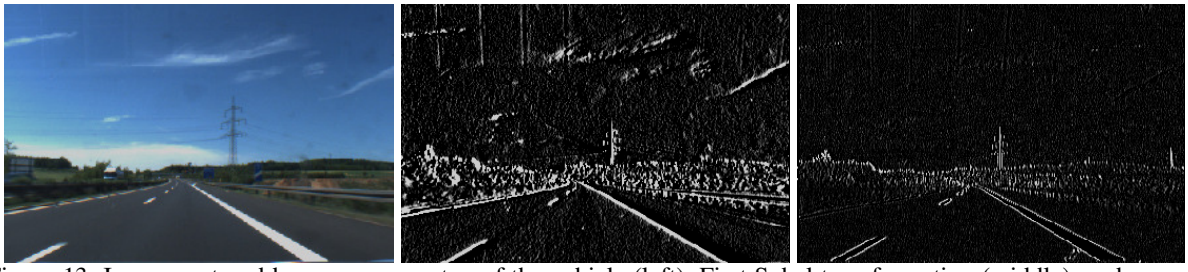

Figure 13: Image captured by a camera on top of the vehicle (left). First Sobel transformation (middle), and second Sobel transformation (right)

Figure 14: Stereo pair with gradient information. To the right we see the disparity map computed from the gradient information in the two images.

We use the information contained in the first and second degree Sobel transformations of the original images to identify the phase of the gradient. Fig. 13 shows an original image, and the result of a first (in the middle) and second degree Sobel transformation (on the right). As can be

seen, each line in the picture has been converted into two white lines. Combining both Sobel transformations, it is possible to identify the white lane lines.

Fig. 14 shows a stereo pair and black-to-white gradients colored in green, while white-to-black gradients have been colored red. We use only the sign, not the absolute value of the Sobel transformation. We obtain a ternary image assigning 1 to green, -1 to red, and 0 to black. Now we can calculate a global disparity map (Fig. 14, to the right) by measuring the line-wise disparity. We start with a ternary vector *L* for the left image and a ternary vector *R* for the right image.

We test each possible disparity by computing the scalar product of both vectors. If a red (green) pixel in the left image matches a red (green) pixel in the right image, the scalar product is increased by 1. This indicates correlation. But if a green (red) pixel in the left image matches a red (green) pixel in the right image, the scalar product is decreased by 1. This decreases the correlation score. In the disparity map the abscissa indicates the disparity, while the intensity (from black to white) indicates the strength of the correlation for this disparity. The map is calculated row by row.

The right image in Fig. 14 shows a distinct line with a certain inclination. This indicates a decreasing disparity as seen from the bottom to the top of the original image. This is due to the fact that the street is oriented towards a vanishing point in the horizon. Vertical objects, such as obstacles, will result in lines parallel to the ordinate in the disparity map. Using a calibrated camera pair, it is even possible to calculate the car's pitch angle by computing the point of intersection between the main line in the disparity map and the ordinate.

We are still developing the stereo vision system in order to better model street markings and detect vehicles on the road. This will be a backup for the laser scanning system.

4.4 Special-purpose vision with a road scanning LIDAR

Scanning white lane markers

One method for road scanning used by some groups is to point a LIDAR towards the ground and work with the reflected intensities. Usually, white lanes are clearly visible. The data is not corrupted by shadows and can be used at night. This method provides a special-purpose computer vision for lane marking, which we have also implemented.

We mounted a SICK LMS 291-S05 scanner in the back of our vehicle. The reflected intensities are arranged in a histogram which exhibits a bimodal distribution. One Gaussian mode represents the range of reflected intensities of the background (the road), the other Gaussian mode, the range of intensities of the white lanes. Using a variation of Otsu's algorithm [Otsu 79] (which finds a point in the middle of the two Gaussian peaks) we compute a dynamic threshold for segmenting the background from the white lanes. While Otsu's method selects a threshold based on the minimization of the within-group variance of two classes obtained by thresholding, our methods selects a threshold by maximizing a within-group quality-measure. We define the total quality of a threshold as the sum of the quality of the points in the left cut plus the quality of the points on the right cut. The quality Q of a set of intensity values $x_1,\ldots,x_n$ (on each side of the cut) is given by

$$Q = \frac{1}{n}\left(\left(\sum x_i\right)^2 - \sum x_i^2\right)$$

Using this threshold to decide whether a ray hits a white lane line is vulnerable to noise (specially when the road material is bright). Therefore, a particle filter is used to track the correct position of the lane markers across several consecutive scans. The particle filter is updated using Sampling Importance Resampling (SIR).

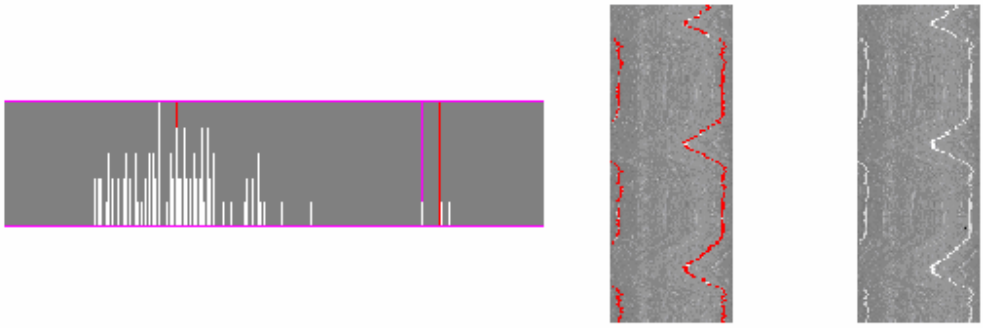

Figure 15: To the right, the histogram of laser intensity values. There are two clusters of intensity values. In the middle, a sequence of laser scans (arranged one of top of another) and the white lane markers found (colored red). To the left we see the corresponding sequence of intensity values of the scans. The car was balancing from one side to the other when the scans were made.

Fig. 15 shows results obtained with the LIDAR pointed to the ground. Even in the extreme case of the car balancing from one side to the other (as in this sequence of images), our method is able to recover the position of the white lines. The main issue with our technique is the number of scan rays which actually hit the lane markers, specially when they are thin. The particle filter solves this problem and produces accurate results.

5 Control

5.1 Low-level Control

For steering, we use a non-linear PD-Controller similar to the one described in [Hoffmann 07]. Given a trajectory (Fig. 16), the distance e of the guiding wheels to the nearest segment of the trajectory and the angle ψ of those wheels with respect to that segment are determined. The formula for computing the output (the steering angle of the leading wheels) includes also the vehicle's current speed v and the constants $k_1, k_2,$ and k_3 determined by matching experimental and simulation results, as we have done with other robotic platforms [Gloye 04].

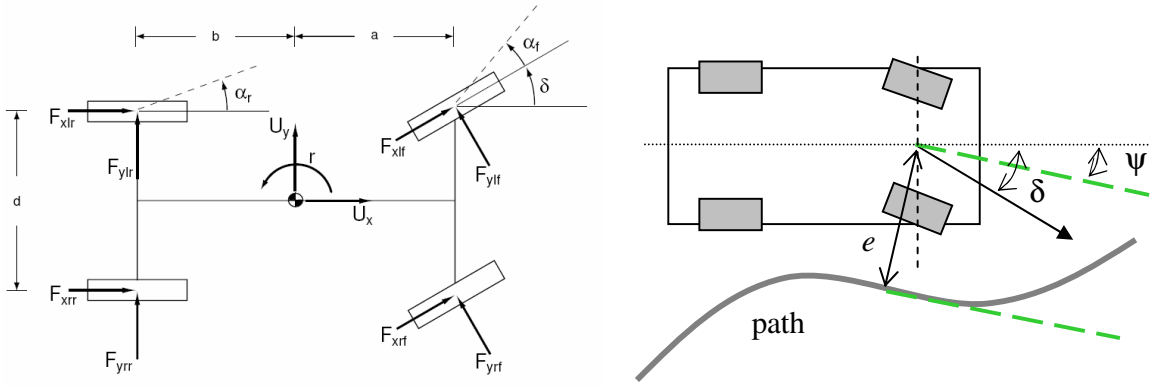

Figure 16: Left image: the car dynamical model as described by [Rossetter 03]. Right image: the angles and variables used for steering control following a desired trajectory (see main text) as described in [Hoffmann 07].

The formula derived in [Hoffmann 07] for steering control is:

Steering control

$$\delta = \psi + \arctan\left(\frac{k_1 e}{k_3 + v}\right) + k_2\left(\delta(i) - \delta(i+1)\right)$$

The delay between commanding and reaching the desired steering angle can cause oscillations. The term $k_2\left(\delta(i) - \delta(i+1)\right)$ is added to the steering command to prevent this condition, where δ is the discrete time measurement of the steering wheel angle and *i* is the index of the discrete measurement one control period earlier. The constant k_3 is included in the expression to obtain better performance at low speeds, and to prevent the argument of arctan from becoming too large. Another way of handling the delay is by using a control prediction approach as we have done in the past for our mobile robots [Behnke 04].

Speed control

The values needed for speed control are contained in the drivable path provided by the mission planner. Each point in the trajectory includes the desired speed and the time at which the vehicle should arrive. We have written a fuzzy PI-controller for reaching the desired values. An output command between -1 (full braking) and +1 (full acceleration) is computed using the difference between the required and current speed (proportional control), and the difference between the required and current position of the vehicle (integral control). If the vehicle is too far away from the desired position, the values in the trajectory are modified in order to avoid control overshooting. I-Control is only possible if the current position of the vehicle does not require acceleration, otherwise problems could arise when the software tries to save time accelerating in curves where we want low speeds. The fuzzy controller is now in use since the original PI-controller could not achieve satisfactory results in our driving experiments.

5.2 High-level control

A mission is defined over a graph of nodes specified in the *Route Network Definition File*. We represent the topology of the graph using the Boost Graph Library, which organizes the RNDF as a weighted directed graph represented by an adjacency list. The edges of the graph represent portions of the lanes and connect road points among themselves, and with input and output gates at crossings. Lane changes and passing of cars are computed using this data structure. The mission contained in the *Mission Data File* is transformed into a sequence of drivable nodes from start to finish using the A* algorithm, very popular for this kind of informed search tasks [Hart 68].

The basis of behavior control is a route planner which computes a default route, which is then adjusted according to the world model. The adjustment is done by finding dynamically a sequence of maneuvers which yield the desired outcome. While computing a path, intermediate results are checked for validity according to the Urban Challenge rules. While the car is driving, we inspect the future trajectory within a time window in "front" of the vehicle; that is, in its future. The process checks for rule conformity using a set of tests which can be extended arbitrarily. If a conflict is found, the critical region in the path is passed to error control which starts a revision of the intended route. Examples of the checks which have been implemented are:

- Obstacle checker: tests future collisions with cars or obstacles. The obligatory safety distances are maintained.

Rules checker

- Speed checker: compares the current speed with the mandatory limits in the segment (minimum and maximum allowed speed). The lateral acceleration in curves is checked against a threshold.
- Lane checker: tests if the car keeps its position within the lane, except when passing or u-turning.
- Crossway checker: tests if the vehicle is within a safety area and adjusts the separation to the forward vehicle.
- Lane error handler: handles the rules A.10, A.11 of the DARPA basic navigation rules brochure.

Fig. 17 shows an example of the Obstacle Checker in action. On the left, a collision with a static object has been predicted for the current path within the next control cycles. Obstacle control finds a smooth path around the static obstacle. In the case of a dynamic obstacle, the car could change lane (in a two lane street), and wait for the opportunity of safely returning to the right lane. This maneuver depends on several circumstances, such as the speed of the dynamic obstacle and the proximity of the next crossing.

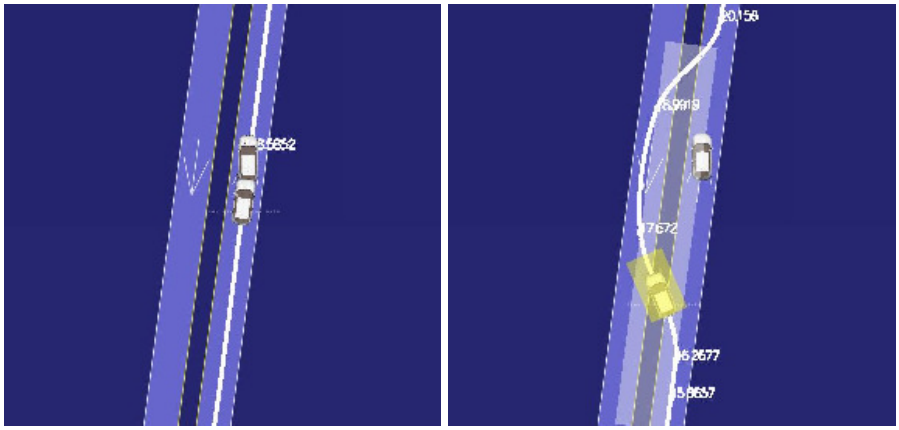

Figure 17: Detection of a future collision and modification of the path so that the collision is avoided

After the planner has found a sequence of drivable nodes in the RNDF, we compute a smooth path connecting these nodes. Lacking information about the real shape of the street, we first approximate the route using an Akima spline, a variation of natural splines, whose building blocks are cubic splines subject to special boundary conditions.

The spline provides a mapping from time to position, providing implicitly information about the desired velocities and accelerations. The Akima spline is useful because it provides continuous transitions up to the first derivative of the acceleration. The main advantage of this interpolation is that local information plays a larger role in the interpolation, reducing the effect of global landmarks. We compute the rate of change at the interface between two intervals, using a heuristic function. For each interval we compute a Hermitian interpolation, so that in each interval only six points (three to the left and three to the right) have an effect on the computation of the polynomial.

The behavior controller manages also the state of the car using a finite state machine. We have defined a set of states in which the car can be found. Transitions are checked for rule compliance and safety.

III Results and Performance

6 Summary of results

In the previous sections we have discussed the rationale for certain design decisions as well as the hardware and algorithms used. Many examples of the results obtained with the different sensors and algorithms have already been provided. A summary follows.

6.1 Navigation and behaviour control results

Fig. 18 shows Spirit of Berlin navigating a closed track in our test area in Berlin (the sequence runs from left to right, top to bottom). The test route consists of two lanes, delimited by white lines. The area covered by the frontal laser scanner is delimited by the two red lines coming out of the vehicles front. Obstacles are marked using bright colored pixels. The colors correspond to the four beams of the Ibeo LIDAR. As can be seen in the figure, the car recognizes an obstacle when it is coming out of the curve and adjusts its trajectory accordingly.

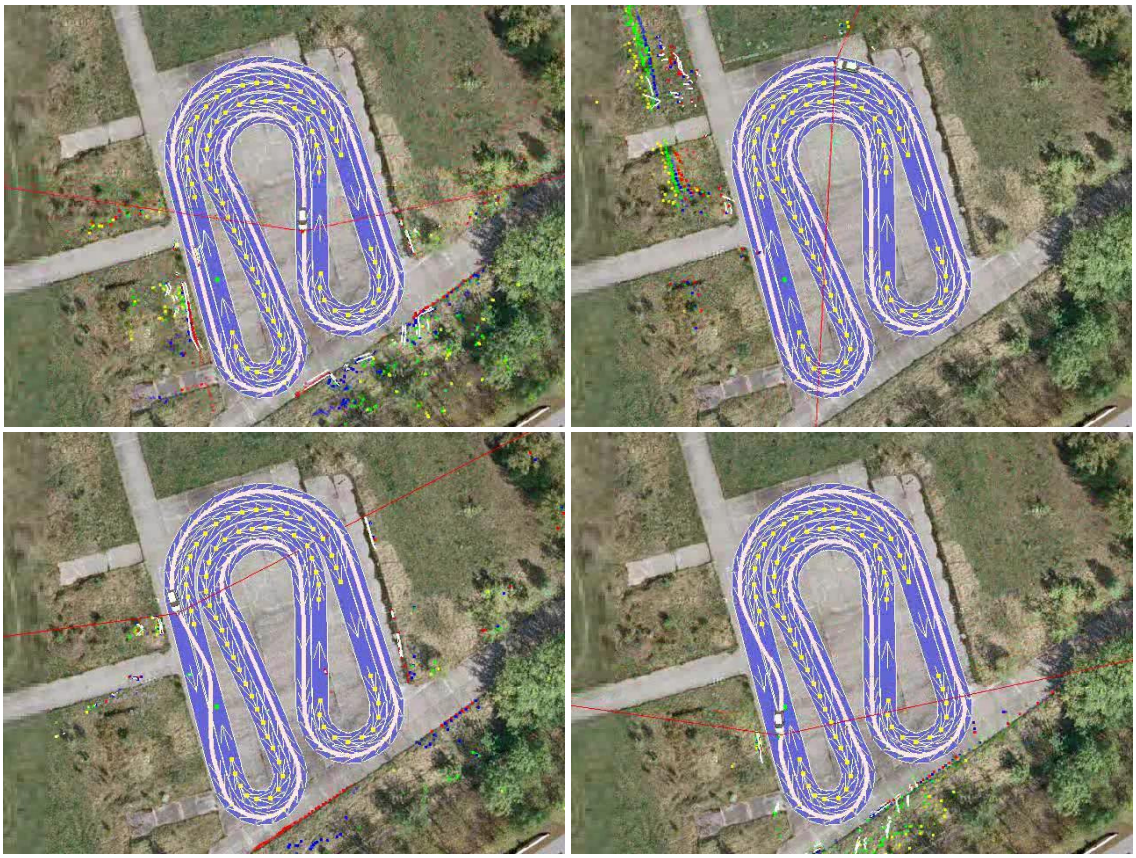

Figure 18: Our autonomous car navigating in a closed loop (sequence from left to right, top to bottom). The red lines show the coverage of the frontal laser scanner (220 degrees). When the car comes out of a curve it detects an obstacle and modifies its trajectory accordingly.

Spirit of Berlin can currently navigate with a mean localization error of less than 30 cm. The adjustments to the trajectory described in this paper allow the car to keep its position within the

lane, overtake a stationary vehicle, and return to its lane safely. Behavior control works in the intended way. We have been driving at moderate velocities (under 12 mph) and will be increasing gradually the speed of the car until we reach Urban Challenge speeds.

6.2 Obstacle detection and world map

The particle filter described above uses the information stored in a world map and has been tested with n poses of the car (n=256 proved to be adequate). Poses with a large matching error are eliminated and new poses are generated. The n poses are divided into three groups: one half are distributed around the current GPS car-position, one fourth are distributed around the best poses, one fourth are distributed around the current mean of the GPS correction offset. A Gaussian distribution with different variances is used. Around current GPS car-position the variance is usually within 0.3-1.0 meters; around the best poses the variance is about 1 cm; around the current offset mean, the variance is usually set to 0.1 meters.

Every matching-step returns an offset for the current frame (given current laser data and GPS-position). In order to minimize noise an average over the last m poses is computed and is used as the actual correction offset vector added to the current GPS-position. The GPS drift can only grow slowly over time, so the offsets should not change too rapidly from one frame to the other.

Test runs on two locations have shown a matching error of up to 30 cm with an average of five to ten cm. This is an improvement over the GPS car position, which may contain an error of up to 1 meter. Fig. 19 shows three images of the world map and the car's position derived from the particle filter (white rectangle), superimposed on the GPS/IMU position. The particle filter allows our system to eliminate GPS drift.

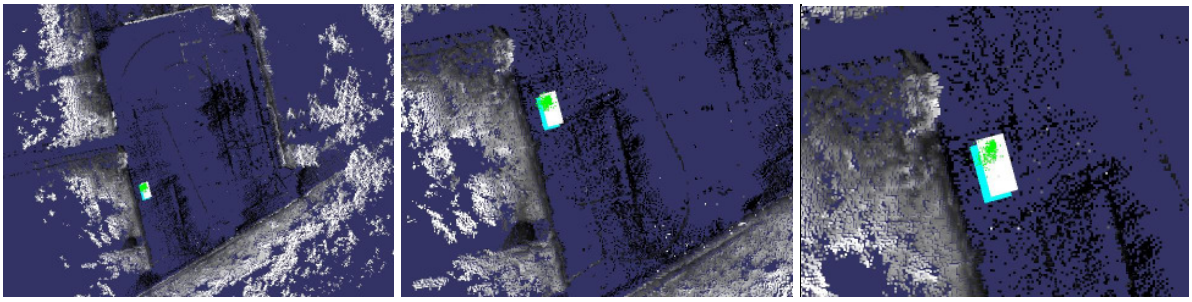

Figure 19: The particle filter in action. The white rectangle is the best car position derived from the mean of the particles. It is superimposed on the rectangle representing the current GPS/IMU position.

6.3 3D scanner segmentation results

The Fraunhofer 3D scanner can be used to recognize objects and textures. Currently, we are working on vehicle recognition. Street recognition has been already implemented. For this, we use the distance and intensity values from the scans. We fuse this information in our road segmentation algorithm to add more robustness to the recognition task.

First, we consider each scan plane separately. In a window of n scan points (i.e. n=4), we calculate the best linear fit and the maximum difference between the intensity of all points in the window. The points do not belong to the road if: a) the linear fit has a large error, b) the linear fit has a large slope or distance relative to the ground plane, c) intensity values are very different. We displace the window, increasing the scan index for each scan plane.

Conversely, if the linear fit error and the change of intensity are lower than selected thresholds, the analyzed segment is probably a road. Values between the minimum and maximum thresholds are an indication of a possible end of the road. Values higher than the maximum threshold are considered as a strong indication of a possible end of the road. After obtaining the segmentation for each scan, we apply neighborhood analysis on the road points to reconstruct the road surface. Fig. 20 shows the results of our road segmentation analysis of the 3D point cloud.

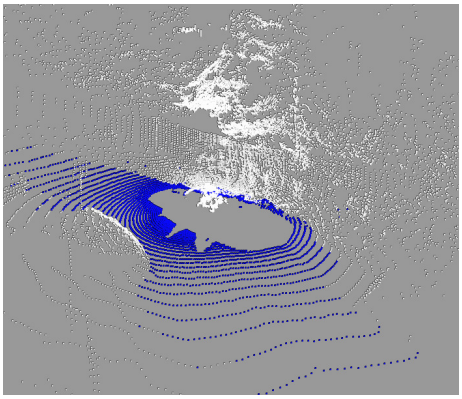

Figure 20: Reconstructed 3D cloud with the detected road points colored blue

The entire 3D cloud, with the segmented road, and a 2D map with the borders of the road are sent to the control computers via Ethernet. We are currently working on a probabilistic road model, as well as SLAM and scan matching techniques for the 3D data.

6.3 Computer vision

Fig. 21 shows the result of processing images at 19 fps. The white lane markers can be identified. They provide additional information for keeping the car safely within its lane. Together with the special purpose laser scanning for lane markers described before, this lane information allows the autonomous car to keep driving safely inside its lane. Driving corrections are integrated with the navigation module in order to produce a smooth control signal.

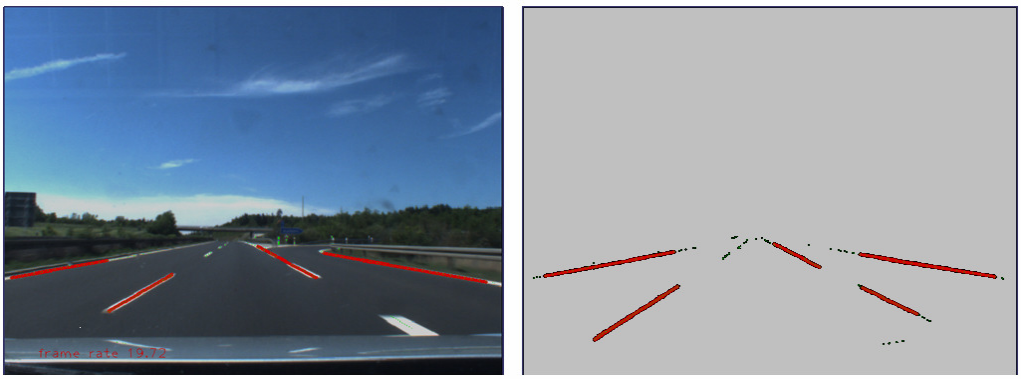

Figure 21: Two examples of lane marker detection. On the right, the lane markers have been highlighted in red. On the left, a more abstract visualization for a subsequent frame. The vision is running at 19 fps.

Acknowledgments

Spirit of Berlin was developed with funding from Freie Universität Berlin, and gifts from IBM Germany and the Microsoft Academic Alliance. The Applanix and Ibeo corporations granted us special Urban Challenge prices for their products. Rice University, Houston, and Southwest Research Institute in San Antonio are hosting Team Berlin during the summer and fall of 2007. Deutsche Forschungsgemeinschaft (DFG) is financing three PhD students who are leading their own development teams within our larger project. Our thanks also to Sebastian Thrun and the Stanford Racing Team for helping to bootstrap our project. Videos of our car driving autonomously can be found in our website at www.spiritofberlin.eu.

References

[Behnke 00] S. Behnke, B.Frötschl, R. Rojas, P. Ackers, W. Lindstrot, M. de Melo, M. Preier, A. Schebesch, M. Simon, M. Sprengel, and O. Tenchio, "Using hierarchical dynamical systems to control reactive behavior," Proceedings IJCAI'99 - *International Joint Conference on Artificial Intelligence*, Stockholm, pp. 28-33, 1999.

[Behnke 04] Sven Behnke, Anna Egorova, Alexander Gloye, Raúl Rojas, and Mark Simon: "Predicting away the Delay", in D. Polani, B. Browning, A. Bonarini, K. Yoshida (eds.): *RoboCup-2003 - Robot Soccer World Cup VII*, Springer, 2004.

[Broggi 05] A. Broggi, C. Caraffi, R. I. Fedriga, and P. Grisleri, "Obstacle Detection with Stereo Vision for off-road Vehicle Navigation," in International IEEE Workshop on Machine Vision for Intelligent Vehicles, San Diego, USA, June 2005.

[Bruyninckx 07] H. Bruyninckx, P. Soetens, "The Orocos Project – Open Robot Control Software", Revision 1.0.2, Technical Report, Orocos Organization, 2007.

[Dahlkamp 06] H. Dahlkamp, A. Kaehler, D. Stavens, S. Thrun, and G. Bradski, "Self-supervised Monocular Road Detection in Desert Terrain", Technical report, Stanford University, 2006.

[Fischler 81] M. A. Fischler, R. C. Bolles, „Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography", Comm. of the ACM, Vol 24, pp 381-395, 1981.

[Gloye 04] Alexander Gloye, Cüneyt Göktekin, Anna Egorova, Oliver Tenchio, Raúl Rojas, "Learning to Drive and Simulate Autonomous Mobile Robots", in D. Nardi, M. Riedmiller, C. Sammut, J. Santos-Victor, J. (eds.): *RoboCup-2004 - Robot Soccer World Cup VIII*, Springer-Verlag, 2004.

[Hart 68] P. E. Hart, N. J. Nilsson, B. Raphael, "A Formal Basis for the Heuristic Determination of Minimum Cost Paths", *IEEE Transactions on Systems Science and Cybernetics* SSC4 (2), pp. 100-107, 1968

[Hundelshausen 01] Felix v. Hundelshausen, Sven Behnke, Raul Rojas, "An omnidirectional vision system that finds and tracks color edges and blobs", in: A. Birk, S. Coradeschi, and S. Tadokoro (editors): *RoboCup-2001: Robot Soccer World Cup V,* Springer-Verlag.

[Hoffmann 07] G. Hoffmann, C. Tomlin, M. Montemerlo, S. Thrun, "Autonomous Automobile Trajectory Tracking for Offroad Driving: Controller Design, Experimental Validation and Racing", American Control Conference 2007, New York, NY, 2007.

[Otsu 79] Otsu, N., "A threshold selection method fom gray-level histograms", IEEE Trans. Syst., Man and Cybern., v. SMC-9:62-66

[Rossetter 03] E. Rossetter, *A potential field framework for active vehicle lane keeping assistance*, PhD Thesis, Stanford University, 2003.

[Scherzinger] B. Scherzinger, "Precise Robust Positioning with Inertial/GPS RTK", White Paper, Applanix Corporation.

[Simon 04] Mark Simon, Fabian Wiesel, Anna Egorova, Alexander Gloye, Raúl Rojas, "Plug & Play: Fast Automatic Geometry and Color Calibration for Tracking Mobile Robots", in D. Nardi, M. Riedmiller, C. Sammut, J. Santos-Victor, J. (eds.): RoboCup-2004 - Robot Soccer World Cup VIII, Springer-Verlag, 2004.

[Thrun 05] S. Thrun et al, "Stanley: The Robot That Won The DARPA Grand Challenge", Technical Report, Stanford University, Department of Computer Science, 2005.